Simple Transliteration for CLIR.

Sauparna Palchowdhury[1] and Prasenjit Majumder[2]

[1] CVPR Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700108, India
sauparna.palchowdhury@gmail.com

[2] Computer Science & Engineering,
Dhirubhai Ambani Institute of Information and Communication Technology,
Gandhinagar 382007, India
p_majumder@daiict.ac.in

Abstract. This is an experiment in cross-lingual information retrieval for Indian languages, in a resource-poor situation. We use a simple grapheme-to-grapheme transliteration technique to transliterate parallel query-text between three morphologically similar Indian languages and compare the cross-lingual and mono-lingual performance. Where a state of the art system like the Google Translation tool performs roughly in the range of 60-90%, our transliteration technique achieves 20-60% of the mono-lingual performance. Though the figures are not impressive, we argue that in situations where linguistic resources are scarce, to the point of being non-existent, this can be a starting point of engineering retrieval effectiveness.

1 Introduction

This is an experiment in cross-lingual information retrieval for Indian languages, in a resource-poor situation. We use a simple grapheme-to-grapheme transliteration technique to transliterate parallel query-text between three morphologically similar Indian languages and compare the cross-lingual and mono-lingual performance. Where a state of the art system like the Google Translation tool[3] performs roughly in the range of 60-90%, our transliteration technique achieves 20-60% of the mono-lingual performance. Though the figures are not impressive, we argue that in situations where linguistic resources are scarce, to the point of being non-existent, this can be a starting point of engineering retrieval effectiveness.

Bengali, Gujarati and Hindi, the three languages we work with in this experiment, share some of the typical characteristics of Indian languages [1]. They are inflectional[4] and agglutinative[5]. Their writing systems use a phonetic alphabet,

[3] http://translate.google.com/

[4] inflection - In grammar, inflection is the modification of a word for expressing tense, plurality and so on.

[5] agglutinative - Having words derived from combining parts, each with a distinct meaning.

where phonemes[6] map to graphemes[7]. There is an easily identifiable mapping between graphemes across these languages. Exploiting these similarities, we use a *grapheme-to-grapheme, rule-based transliteration* [2] technique. The rules mapping graphemes in the two alphabets are constructed manually.

The manual construction is fairly easy for these three languages because the graphemes in the Unicode chart are arranged in such a way that the similar-sounding entities are at the same offset from the table origin. For example the sound 'k' is the $22^{nd.}$ ($6^{th.}$ row, $2^{nd.}$ column) grapheme in all the three languages, and one distinct grapheme represents 'k' in each language.

Two issues in CLIR is tackling synonymy[8] and polysemy[9]. Translation addresses these issues, but it needs language resources like dictionaries, thesauri, parallel corpora and comparable corpora. On the other hand transliteration is able to move the important determinants in a query like out-of-vocabulary (OOV) words and named-entities (NE), across languages, fairly smoothly.

We retrieve from our collections using the original query and its translated (using the web-based Google Translation tool) and transliterated versions, and compare the performance in the rest of the paper. The Section 2 places our work in context, describing the related work in Indian Language IR (ILIR). Section 3 briefly mentions our benchmark collections. A detailed description of the experiments is in Section 4. Our transliteration technique is explained there. The results are discussed in Section 5. We close our exposition with conclusions, limitations and suggestions for future work in Section 6.

2 Related Work

Transliteration of query-text to a target language is an important method for cross-language retrieval in Indian languages because language resources are scarce, and transliteration can move NEs and OOV words fairly smoothly from one language to another. NEs and OOV words being important determinants of information-need in many queries, protecting them from distortion helps improve retrieval effectiveness. A common next-step to transliteration is fixing the defective NEs and OOV words. ILIR has recently been evaluated by the *Forum for Information Retrieval Evaluation*[10], where several transliteration techniques were tried ([3], [4]). Kumaran et al. [5] tries combining several machine transliteration modules. They use English, Hindi, Marathi and Kannada, and leverage a state-of-the art machine transliteration framework in English. Chinnakotla et al. [2] applies a rule-based transliteration technique using Character Sequence

[6] phoneme - A phoneme is the indivisible unit of sound in a given language. It is an abstraction of the physical speech sounds and may encompass several different phones.

[7] grapheme - The smallest semantically distinguishing unit in a written language. Alphabetic letters, numeric digits, punctuations are examples of graphemes.

[8] synonymy - Being synonymous; having same meaning.

[9] polysemy - A word having multiple meanings.

[10] www.isical.ac.in/~fire

Modelling (CSM) to English-Hindi, Hindi-English and Persian-English pairs. Our work is an empirical approach, focusing on a few Indian languages that share similar syntax, morphology and writing systems.

3 Benchmark Collection

The test collection[11] we used is the latest offering of the 3$^{rd.}$ FIRE workshop held in 2011. We used the Bengali, Gujarati and Hindi collections, and all the 50 queries in each of these languages. The queries were formulated from an information-need expressed in English and translated to six Indian languages by human translators.

4 Retrieval Runs

At the outset we describe the entire procedure in brief. We worked with Bengali (*bn*), Gujarati (*gu*) and Hindi (*hi*). We set up retrieval runs over several variations of the indexed test collections and the queries, using Terrier-3.5 [6]. The resources at hand were the test collections, queries, qrels, stop-word lists and stemmed word-lists for the three languages. We used the statistical stemmer; YASS [7].

Referring to the graphical representation of the experiment in Figure 1, Table 1, 2 and 3 may help the reader follow the description in this paragraph. Starting with a query in one language (the source language), its text was translated and transliterated to another language (the target language). The transliteration was redone by stopping and stemming the source. Thus each source language text yielded three versions of that text in the target language.

The transliteration technique simply added an offset to the hexadecimal Unicode value of each character in the alphabet. There being no strict one-to-one mapping between graphemes between the source and the target languages, manually defined mappings were used where necessary (explained in Section 4.1 on transliteration).

So, as an example, for Bengali as the target language, we ended up with 3 types of text in Bengali (*bn.gu.g*, *bn.gu* and *bn.gu.p*), sourced from Gujarati (*gu*) and 3 more (*bn.hi.g*, *bn.hi* and *bn.hi.p*) sourced from Hindi (*hi*). The prefix *bn.gu*, is of the form *target.source*, and is suffixed by letters denoting the variations. The absence of the suffix denotes the text obtained by our transliteration technique. The *.g* suffix marks the text as obtained by translation using Google Translation tool, and the *.p* suffix marks the text as obtained by our transliteration technique after pre-processing by stopping and stemming the source. Including the original Bengali query (*bn*), we had 7 (3 + 3 + 1) versions of Bengali query text. Putting all the string in a set we get R = {*bn*, *bn.gu.g*, *bn.gu*, *bn.gu.p*, *bn.hi.g*, *bn.hi*, *bn.hi.p*} for one source-target language pair.

[11] http://www.isical.ac.in/~fire/data.html

For each of the 7 versions in set R, we set up 3 retrieval runs by varying the query processing steps; no-processing or the empty step (e), stopping-and-stemming (sS), and query expansion (x) denoted by the set R1 = {e, sS, x}. Another 2 variations were done for each of these three; one using the topic *title* and another using the *title*-and-*description* fields of the queries, denoted by the set R2 = {T, TD}. Summing it up, we had R X R1 X R2 runs, or, 7 * 3 * 2 = 42 runs for each language. Working with 3 languages, we submitted 42 * 3 = 126 runs at the 3rd. FIRE workshop.

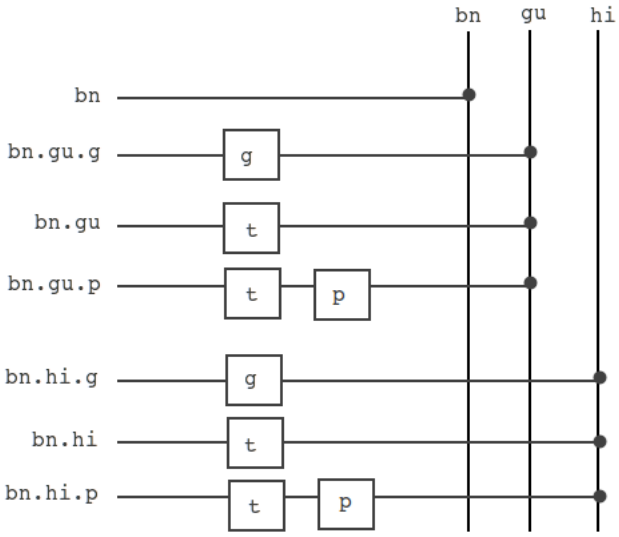

Fig. 1. The way the seven types of Bengali query text were generated. The diagram flows from right to left. The three source languages are at the top right, and lines tapped from them lead to the target language versions on the left. The *bn.gu* prefix denotes a *target.source* language pair. g is the Google Translation tool, t is our transliteration technique and p chips in as a stopping-and-stemming step of pre-processing before going through t.

4.1 Transliterating Graphemes

In *Unicode Standard 6.0*, 128 code-points are allocated to each Indian language script. The Devanagari script, used for Hindi (henceforth, we use the phrases 'Hindi script' and 'Devanagari script' interchangeably), assigns a grapheme to all code-points except one, whereas the Bengali script has 36, and Gujarati 45, missing points. The relative positions of the phonetically similar letters being identical in the Unicode chart for the three languages, adding and subtracting

1.	*bn* - The original Bengali query text
2.	*bn.gu.g* - Translating Gujarati to Bengali using the Google Translation tool.
3.	*bn.gu* - Transliterating Gujarati to Bengali using our technique.
4.	*bn.gu.p* - Transliterating, after pre-processing by stopping and stemming the query text.
5.	*bn.hi.g* - Type 2 using Hindi as the source language.
6.	*bn.hi* - Type 3 using Hindi.
7.	*bn.hi.p* - Type 4 using Hindi.

Table 1. Set R. The seven types of Bengali query text. The strings are best read off from right to left.

1.	*e* - No processing whatsoever, query and document text remains as it is.
2.	*sS* - Stop-words removed and remaining words were stemmed using YASS.
3.	*x* - Query expansion (Terrier-3.5's default; Bo1).

Table 2. Set R1. Three ways of retrieval. *e* is the no-processing or the empty step. The *sS* step needs the collection to be indexed with stopping and stemming enabled. For *x* we use the stopped and stemmed index.

1.	*T* - Retrieval using only the *title* of a query.
2.	*TD* - Retrieval using the *title* and *description* fields of a query.

Table 3. Set R2. Two more ways of retrieval, using the *title* and *description* fields of the queries.

hex offsets worked for most cases but for the missing code-points. We had to take care of many-to-one mappings (which occurred frequently when mapping Hindi to the other scripts) and mapping letters to NULL (which was equivalent to ignoring them), when a suitable counterpart was not found. Here is how we handled such situations, described for the reader who has some familirity with Indian scripts.

(a) When a grapheme had no counterpart in the target language: Devanagari vowel sign OE (0x093A) was mapped to NULL (0x0). Bengali AU (0x09D7) length mark was mapped to NULL.
(b) When a grapheme had a phonetically similar counterpart: Devanagari short A (0x0904) was mapped to A in Bengali (0x0985) and Gujarati (0x0A85).

Gujarati LLA (0x0AB3) was mapped to Bengali LA (0x09B2). Hindi has a LLA (0x0933) too.

(c) When a grapheme's usage changed in the target language: ANUSVARA (for a nasal intonation) is used independently in Bengali (0x0982), but in Hindi (0x0902) it almost always resides as a dot on top of a consonant and results in pronouncing N, so it was mapped to Bengali NA (0x09A8).

(d) VA and YA was correctly assigned. VA is pronounced YA in Bengali. Bengali does not have a VA. Whereas YA in Bengali is YA in the other two languages, and not YYA, which also exists.

All in all we had to manually map 18 Bengali, 8 Gujarati and 50 Hindi graphemes, as shifting by hex offsets would not work for them. The transliteration program may be download from a public repository[12].

5 Results and Analysis

The results show all the 126 runs in Figure 2 and Table 4. The bar charts give us a quick visual comparison of the runs. Our baseline is the mono-lingual run using the original query (the leftmost bars in each of the seven stacks in each chart). It is the best possible performance in the current set-up. The output of the Google Translation tool is our cross-lingual baseline. It is a state of the art tool which is expected to have made use of language resources, helping us compare to it our resource-poor methods.

The retrieval runs show improved performance in the increasing order $e < sS < x$, and $T < TD$. Oddly, for *gu.hi* $T > TD$.

Therefore *x-TD* retrieval runs (retrieval with query expansion and the *title-and-description* fields) are the best results amongst all the runs. Query text translated using the Google Translation tool performs over a wide range, from 59-87% of the mono-lingual performance. In comparison our transliteration technique's performance ranges from 19-60%.

The pre-processing step of stopping and stemming the query text before transliterating them does not seem to provide any benefit. It was surmised that stopping and stemming the source text would leave behind cleaner text, as input to the translieration step, by removing the large number of inflections, but this is not corroborated by the results. YASS being a statistical stemmer, tuning it to vary its output, could well be a way to experiment further with the pre-processing.

Gujarati and Hindi seem to be morphologically closer in that the performance of queries across these two languages are better than the cases where Bengali is involved.

A per-query view of our results, in Figures 3 to 8, show how conversion between Bengali and the other two languages have not produced good results. The Bengali charts are significantly sparse, as many queries simply failed to retrieve enough relevant documents. Gujarati-Hindi conversion have been relatively better.

[12] https://bitbucket.org/sauparna/irtools/src/26e3bed0e338/mapchar.c

Bengali	T			TD		
Query type	e	sS	x	e	sS	x
bn	0.2218	0.2538	0.2910	0.2744	0.3242	0.3704
bn.gu.g	59	60	61	60	60	62
bn.gu	23	21	23	23	26	29
bn.gu.p	22	20	21	21	22	22
bn.hi.g	66	63	60	62	59	60
bn.hi	22	25	27	22	28	31
bn.hi.p	10	23	25	11	20	25
Gujarati	T			TD		
Query type	e	sS	x	e	sS	x
gu	0.2236	0.2578	0.2797	0.2611	0.2860	0.3095
gu.bn.g	64	64	64	67	69	68
gu.bn	19	20	27	20	25	30
gu.bn.p	15	18	22	17	23	29
gu.hi.g	65	67	67	72	74	77
gu.hi	54	53	60	28	30	28
gu.hi.p	25	32	37	12	17	17
Hindi	T			TD		
Query type	e	sS	x	e	sS	x
hi	0.1442	0.1532	0.1706	0.1631	0.1750	0.1877
hi.bn.g	59	59	59	69	70	76
hi.bn	19	21	19	24	23	27
hi.bn.p	18	29	31	20	32	37
hi.gu.g	65	71	71	82	83	87
hi.gu	44	42	48	49	49	53
hi.gu.p	39	39	43	42	42	49

Table 4. The comparison of runs in terms of percentage of the mono-lingual performance. The first row of each block is the MAP value for the monolingual run. For a description of the run types refer to Table 1. The rest of the values are % of the monolingual MAP. For example, at row *hi.gu.g*, which denotes retrieval using the query translated from *gu* to *hi* using the Google Translation tool, and column *TD* and *x*, the performance is 87% of the mono-lingual Hindi run. The transliteration technique denoted by *hi.gu* in the same column but in the next row, makes it to 53%. Note that our pre-processing step does not turn out to be useful.

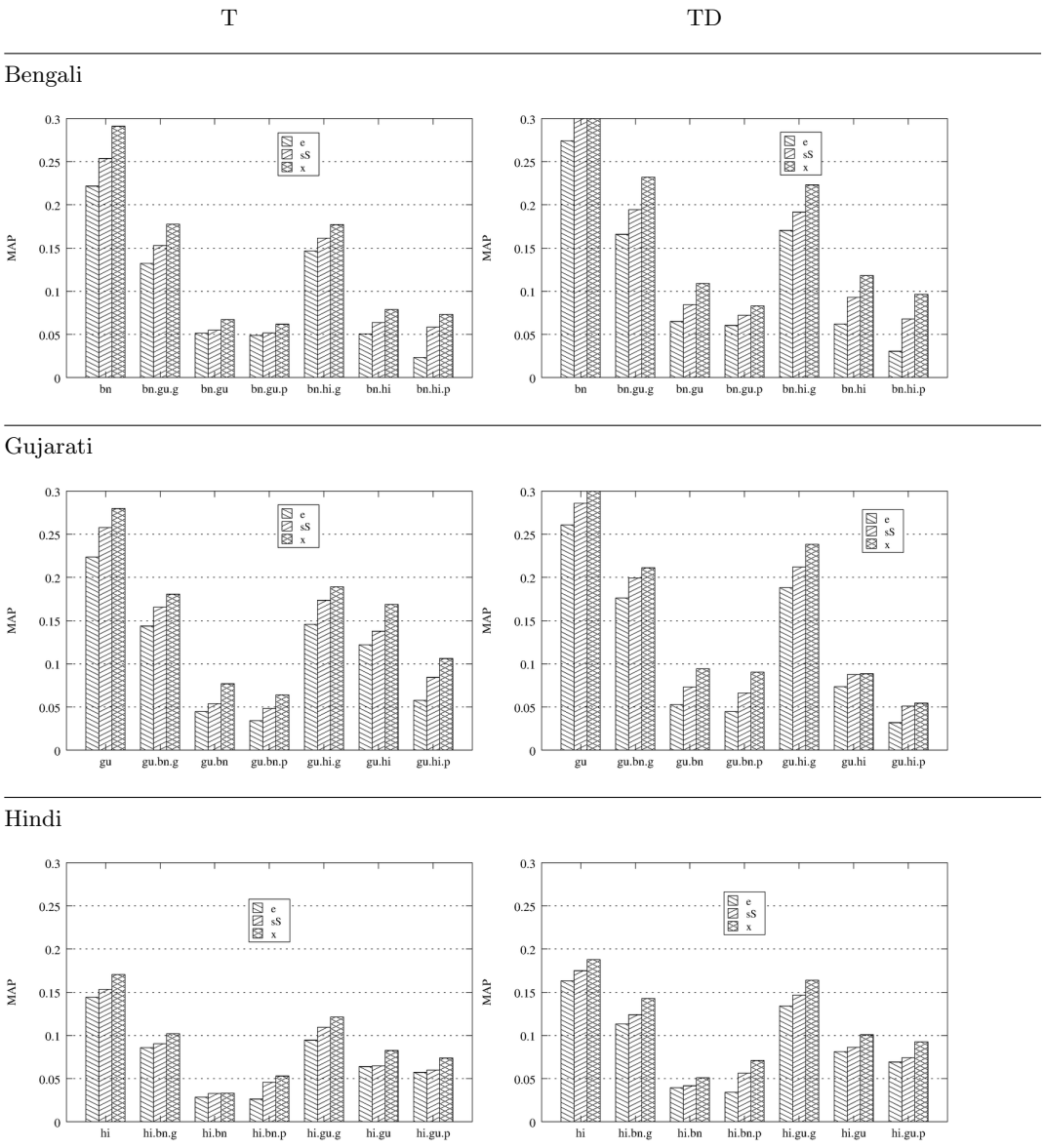

Fig. 2. The six charts show the MAP values obtained for the seven kinds of retrieval runs. Column 1 and 2 is for the T and TD runs, and a row each for the languages Bengali, Gujarati and Hindi. And in each stack of three bars in each chart, the left-to-right ordering of the patterned bars corresponds to the elements of the set R1 = {e, sS, x} in order.

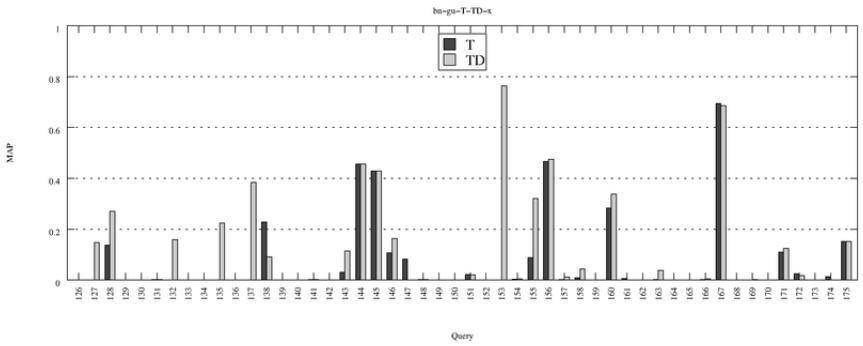

Fig. 3. bn.gu

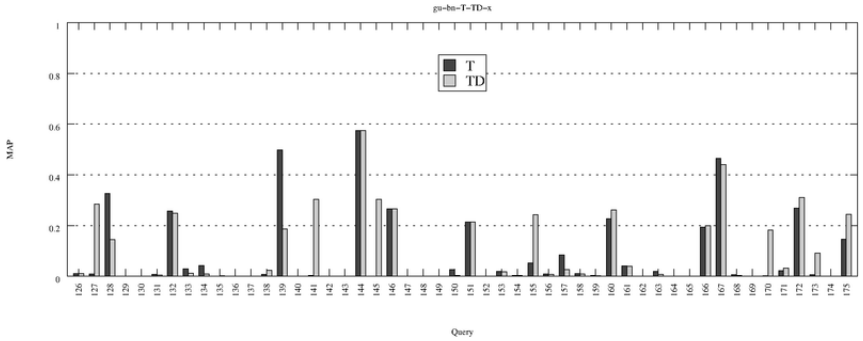

Fig. 4. gu.bn

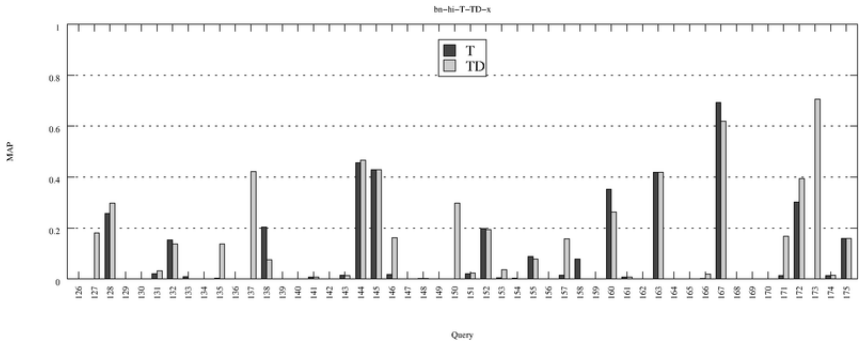

Fig. 5. bn.hi

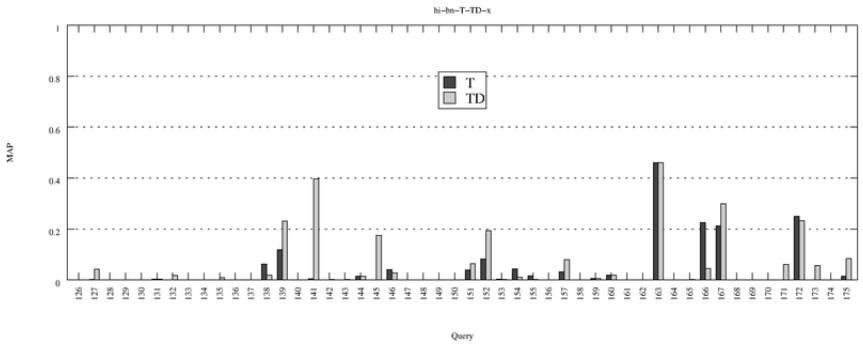

Fig. 6. hi.bn

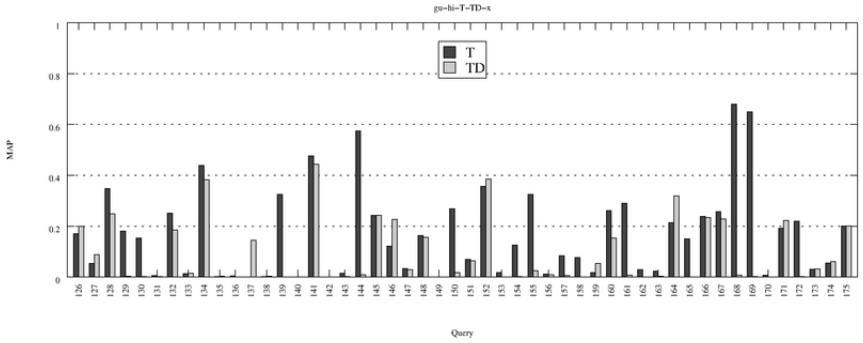

Fig. 7. gu.hi

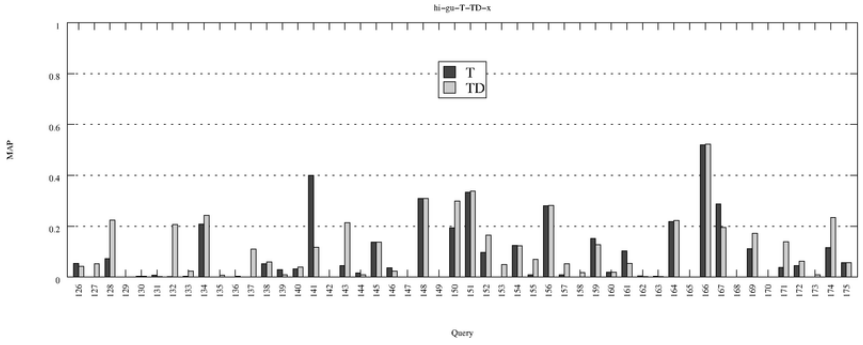

Fig. 8. hi.gu

6 Conclusion and Future Work

We have made an attempt to make use of the similarity in the scripts of a group of Indian languages to see how retrieval performance is affected. It could well have worked for any pair of language, sharing these traits, whose graphemes could be assigned a mapping manually. There are deficiencies in our methods, for example, we have not taken care of spelling variations. A spelling using I (simple 'i') in one Indian language may use II (stressed 'i') in stead. The words of same meaning, which are completely differently spelt in two languages are sure to affect the performance. For example 'vaccine' is 'tika' (English transliteration) in Bengali and Hindi, but 'rasi' (English transliteration) in Gujarati. Only a dictionary could resolve such differences. One other resource that we have not exploited in this experiment is the test collection itself. The noisy converted texts may be augmented in some way by picking evidence from the vocabulary of the test collections. An approximate string matching between noisy query words and the words in the vocabulary could be helpful in identifying the unaltered counterpart with some degree of accuracy and add or substitute them in the query text to improve it.

References

1. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Pal, S., Modak, D., Sanyal, S.: The fire 2008 evaluation exercise. Proceedings of the First Workshop of the Forum for Information Retrieval Evaluation, 2008. **9**(3) (September 2010) 10:1–10:24
2. Chinnakotla, M.K., Damani, O.P., Satoskar, A.: Transliteration for resource-scarce languages. ACM Trans. Asian Lang. Inf. Process **9**(4) (2010) 14
3. ACM Transactions on Asian Language Information Processing (TALIP) **9**(3) (2010)
4. ACM Transactions on Asian Language Information Processing (TALIP) **9**(4) (2010)
5. Kumaran, A., Khapra, M.M., Bhattacharyya, P.: Compositional machine transliteration. ACM Trans. Asian Lang. Inf. Process **9**(4) (2010) 13
6. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A High Performance and Scalable Information Retrieval Platform. In: Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006). (2006)
7. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet another suffix stripper. ACM Trans. Inf. Syst **25**(4) (2007)